REMARKS

Applicant respectfully requests reconsideration and allowance of the subject application. Claims 18-19 are canceled. Claims 1, 16, 17, 24, 27, and 33 are amended. Accordingly, claims 1-17 and 20-37 remain pending.

Applicant thanks the Examiner for the detailed analysis presented in the preceding Office Actions.

CLAIM REJECTIONS UNDER 35 U.S.C. § 103

Claims 1-26, 33, and 37 were rejected under 35 U.S.C. § 103 as being as unpatentable over U.S. Patent No. 6,629,128 B1 to Glass (hereinafter "Glass") in view of U.S. Patent No. 6,560,591 B1 to Memmott et al. (hereinafter "Memmott"). Applicants have carefully considered the reasoning expressed in the preceding Office Actions. Applicants respectfully traverse the rejection, and submit that the claims, as amended, are in condition for allowance.

The subject application is directed to challenges faced in managing systems and devices in an enterprise environment. As computer systems and networks continue to increase in size and complexity, so too does the challenge of managing them. A significant tool that assists network developers and administrators in managing computers across an enterprise is Windows® Management Instrumentation (WMI). WMI enables the remote management of Windows-based systems and applications by exposing management information through an object-oriented structure defined using WMI schemas. WMI schemas are an implementation of the Common Information Model (CIM) as defined by the Desktop Management Task Force (DMTF).

WMI supports the management of systems and devices by exposing management information across an enterprise, such as hardware settings, performance information, driver configurations, BIOS information, application settings, event log information, and so on, and by providing a mechanism to query for information and configure settings on machines across the enterprise. WMI provides access to management information on a single network machine, or a large number of machines all at once. For example, without WMI, an administrator wanting to enumerate descriptions for various groups of objects on a machine must locate and learn different application programming interfaces (APIs) that describe the specific methods for communicating with each group of objects. However, WMI eliminates the need to learn the specifics of every API set provided by Windows, through gathering information from a diverse set of APIs and presenting this information in a simple, industry-standard management object model.

Therefore, many comprehensive and well-documented managed resources are available to those developers and administrators capable of utilizing the benefits of WMI. However, WMI is generally designed for use by developers or administrators who are at least moderately proficient at programming in C/C++, Microsoft Visual Basic®, and scripts.

Writing such scripts, however, may be beyond the ability of many administrators, and the discovery of basic system information may therefore be difficult without the assistance of a more experienced programmer. In addition, much of the power of WMI is realized through developers writing management applications that monitor, configure and control the management information made available through WMI. Therefore, the benefits of WMI are often difficult

to attain for the common administrator who does not have the proper programming background, but who still has a need to manage system components/objects.

Independent claim 1 has been amended, as have the other independent claims. claim 1, as amended, is reproduced below:

1. A command line utility embodied in one or more computer-readable media, the command line utility comprising:

a command schema including one or more commands enabling at least one of retrieval of management information from and initiation of a management service available through an object model target schema recognized by at least one target station accessible over a network;

an interactive user interface configured to receive the one or more commands in the command schema from a user and communicate a response of the at least one target station to the user; and

an object model command schema to define a mapping between the one or more commands in the command schema and the ~~an~~ object model target schema and interpret the one or more commands from ~~generated by~~ the command schema to cause one of the retrieval of management information from and the initiation of the management service on the at least one target ~~and configured to operate against the target schema through the command line utility~~.

Applicants respectfully call attention to the second paragraph, reciting the interactive user interface.

Glass describes a system that facilitates communications between two different programs by creating proxies allowing a client application to communicate with a server-based object:

> According to an embodiment of the present invention, a system for distributed processing in a computer network is provided that includes, a client side object request broker executing on a client computer and a server-side object request broker executing on a server computer. The server computer is connected to the client computer through a network. ***A remote proxy generator dynamically generates remote proxy classes for client-side communications support for communications between a client application and a server object.*** The remote proxy generator resides in the server-side object request broker and instantiates the remote proxy class to create a remote proxy object. A client-side type generator generates a client side type object for a class of the server object. ***The client-side type object provides access to methods of the server object. A client-side function generator generates one or more client-side function objects for providing a connection to one or more methods of the server object.***

(*Glass*, Column 3, Line 66, through Column 2, Line 12; emphasis added).

In this context, it is clear that Glass's description of an "interface" is an interface between programs, not a user interface. The inter-object interface contemplated by Glass is defined in the first paragraph of the background of the invention:

> Classes may also be characterized by their ***interface which defines the elements necessary for proper communication between objects.***

(Glass, Column 1, Lines 29-31; emphasis added). Furthermore, while there a few mentions of users in the background of the invention of Glass (*See Glass*, Column 1, Line 46; Column 3, Lines 54-59), the word "user" is never mentioned in the detailed description of the invention of Glass. Moreover, the phrase "user

interface" is never mentioned at all. Because Glass does not describe a user interface, applicants respectfully submit that Glass neither teaches nor suggest the elements recited in claim 1.

Applicants acknowledge the Office Action's recitation of "a command line predevelopment utility" (*Glass*, Column 19, lines 10-14). Respectfully, however, the "command line predevelopment utility," when considered in context, is not a user interface at all. The "command line predevelopment utility" is an alternative inter-object communication mechanism used by Glass to facilitate communication between software objects that do not include a suitable inter-object interface:

> Referring to FIG. 11, ***an interface generator 250 is illustrated for use in remote enabling classes without interfaces.*** A typical remote proxy 154 resides in client system 102 and communicates through network 106 with server object 110 using server object interface 111. Existing class files on server system 104 may need to be used remotely by client application 108 on client system 102. Before the existing class file may be used remotely, it should have an interface in order to comply with the communication standards of typical ORBs. Interface generator 250 generates an interface 254 for a class file 252. Interfaces provide for inheritance from multiple sources and ease of method invocation. ***Without interfaces, a complex procedure using reflection is used to invoke methods directly on objects.***
>
> ***In one embodiment, interface generator 250 is a command line predevelopment utility used to generate interfaces for classes on server system 104 that will be used remotely in distributed computing system 100. In that embodiment, the software developer knows that certain class files 252 will be used remotely. The software developer provides interface generator 250 with a list of class files 252 for which interfaces 254 are to be generated.***

(*Glass*, Column 18, Line 63, through Column 19, Line 17; emphasis added). This appears to be the only mention of a "command line" included within Glass. It is clear that the "command line" as used in Glass is part of "predevelopment utility" enabling a "software developer" to identify objects for which an inter-object interface will be needed; respectfully, it is just as clear that the "command line

predevelopment utility" is not a "user interface," as recited in claim 1. The only interface taught or suggested by Glass is an interface between program objects. Thus, Glass fails to teach or suggest the subject matter of claim 1.

Applicants note that the other independent claims also have been amended to recite or clarify inclusion of a user interface. A user interface is recited, in pertinent part, in each of the independent claims as recited below:

- Claim 16, as amended:

  an interactive user interface configured to receive one or more commands in the command schema from a user and communicate a response of the at least one target station to the user; and

- Claim 17, as amended:

  an interactive interface utility configured to receive a user command to facilitate implementation of individual commands within the set of commands, wherein the interactive interface utility includes at least one of:

  a command line interface utility configured to receive textual entry of the user command; and

  a graphical user interface utility.

- Claim 24, as amended:

  an interactive interface configured to receive the set of commands from a user, wherein the interactive interface includes at least one of:

  a command line interface utility configured to receive textual entry of the user command; and

  a graphical user interface utility.

- Claim 33, as amended:

providing an interactive user interface configured to receive commands from a user;

Applicants submit that each of claims 16, 17, 24, and 33 recite at least one element that is neither taught nor disclosed by Glass. Thus, applicants request that the rejections under 35 U.S.C. § 103 be withdrawn from claims 1, 16, 17, 24, and 33.

For these same reasons, applicants request that the rejections under 35 U.S.C. § 103 be withdrawn from claims 2-15, 20-23, 25-26, and 34-37. Because dependent claims 2-15, 20-23, 25-26, and 34-37 apply additional limitations to the claims from which they depend, these claims are patentable for at least the same reasons as the claims from which each depends, as previously described. Thus, applicants respectfully request that the rejection under 35 U.S.C. § 103 be withdrawn with regard to 2-15, 20-23, 25-26, and 34-37.

In addition, although applicants do not concede that one of ordinary skill in the art at the time the invention was made would supplement Glass with Memmott, applicants submit that Memmott fails to make up for the shortcomings of Glass. Independent claim 1, in pertinent part, recites mapping of commands to object model target schema:

> [A]n object model command schema to define a mapping between the one or more commands in the command schema and the object model target schema and interpret the one or more commands from the command schema and received via the user interface to cause one of the retrieval of management information from and the initiation of the management service on the at least one target.

Respectfully, Memmott neither teaches nor discloses what is recited in claim 1.

Memmott, in pertinent part, "free[s] a requesting entity from the burden of having to select a particular provider" (*Memmott*, Column 3, Lines 7 and 8). More

particularly, Memmott discloses a "data resolver" that is used to retrieve information for a "data requester," regardless of which "data provider" maintains the requested data (*Memmott*, Column 3, Lines 7-67). Accordingly, Memmott provides access to data without the data requester having to know where the data resides, freeing the data requester from having to manage the information or the services that provide it.

On the other hand, while Memmott frees the data requester from having to know what data provider stores the desired information, Memmott requires the data requester to formulate a request for data in the format recognized by the data provider:

> Possible formats for the query received from data requestor 110 include object-oriented formats such as Managed Object Format (MOF) and syntaxes such as Extensible Markup Language (XML). For example, the query may conform to at least one among the distributed management schemes referenced above (SNMP, CMIP, DMI, CIM) or to a similar scheme such as Windows Management Interface (WMI, Microsoft Corp., Redmond, Wash.). Included in the query is a query characteristic that identifies the information requested and/or the subject matter of the query. For example, a query relating to a DVD (Digital Versatile Disk) drive may include an object class (e.g. MediaAccessDevice), a subclass (e.g. DVDDrive), and an indication of the particular drive property about which information is desired

(*Memmott*, Column 3, Lines 27-42). Thus, while Memmott provides the data requester some freedom in identifying the data provider, the data requester must request data in the manner dictated by the data provider. Accordingly, because Memmott is directed to provider transparency, applicants submit that Memmott fails to redress shortcomings of Glass. Thus, applicants submit that the claims are allowable over Glass in view of Memmott.

Claims 27-32 were rejected under 35 U.S.C. § 103 as being as unpatentable over Memmott in view of Glass and in further view of Steve, "Network and System Management with XML" (hereinafter "Nash").

Independent claim 27 is amended as reproduced below:

27. (Currently amended) A method comprising:

receiving a command from a user through an interactive command line interface;

fetching an alias for the command;

interpreting the command based on the alias and the current operating environment of the command line interface;

executing the command as one or more WMI API calls against a targeted namespace representing at least one target system that is accessible via a network;

receiving WMI data in XML form;

applying an XSL style sheet format the WMI data; and

presenting the WMI data through the command line interface.

Applicants respectfully call attention to the second and third paragraphs, reciting fetching an alias and interpreting the command based on the alias.

As previously described, Memmott requires the data requester to formulate a request for data in the format recognized by the data provider. Although Memmott may provide location transparency, it provides no command interpretation. Moreover, although applicants do not concede that one of ordinary

skill in the art at the time the invention was made would supplement Memmott with Glass, for the reasons described previously, applicants submit that Glass fails to make up for the shortcomings of Memmott. Further, applicants submit that Nash fails to make up for the shortcomings of Memmott or Glass, either alone or, for sake of argument, combined to teach or suggest the subject matter of claim 27. Thus, applicants request that the rejection under 35 U.S.C. § 103 be withdrawn from claim 27.

For these same reasons, applicants request that the rejections under 35 U.S.C. § 103 be withdrawn from claims 28-32. Because dependent claims 28-32 apply additional limitations to the claim from which they depend, these claims are patentable for at least the same reasons as the claims from which each depends, as previously described. Thus, applicants respectfully request that the rejection under 35 U.S.C. § 103 be withdrawn with regard to 28-32.

CONCLUSION

Claims 1-17 and 20-37 are in condition for allowance. Applicant respectfully requests reconsideration and prompt allowance of the subject application. If any issue remains unresolved that would prevent allowance of this case, the Examiner is requested to urgently contact the undersigned attorney to resolve the issue.

Respectfully Submitted,

Date: Aug. 3, 2005

By: _____

Frank J. Bozzo
Lee & Hayes, pllc
Reg. No. 36,756
(206) 315-4001